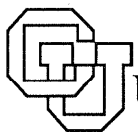
Centroids, Representations and Submodular Flows

Harold N. Gabow

CU-CS-660-93 July 1993



University of Colorado at Boulder
DEPARTMENT OF COMPUTER SCIENCE

Centroids, Representations and Submodular Flows

Harold N. Gabow*
Department of Computer Science
University of Colorado at Boulder
Boulder, CO 80309
hal@cs.colorado.edu

July 28, 1993

Abstract. The notion of the centroid of a tree is generalized to apply to arbitrary intersecting families of sets. Centroids are used to obtain a compact representation for intersecting and crossing families. The size of the representation for a family on n elements is $O(n^2)$, compared to $O(n^3)$ for previous representations. Efficient algorithms to construct the representation are given. For example on a network of n vertices and m edges, the representation of all minimum cuts uses $O(m \log(n^2/m))$ space; it is constructed in $O(nm \log(n^2/m))$ time (this is the best-known time for finding one minimum cut). The representation is used to improve several submodular flow algorithms. For example a minimum-cost dijoin can be found in time $O(n^2 m)$; as a result a minimum-cost planar feedback arc set can be found in time $O(n^3)$. The previous best-known time bounds for these two problems are a factor n larger.

---

* Research supported in part by NSF Grant No. CCR-9215199.

1. Introduction

Consider a tree on n nodes. A *centroid* is a node whose deletion leaves no subtree containing $> n/2$ nodes [WW]. In 1869 Jordan showed that any tree has a centroid (see [H]).

We generalize Jordan's result to certain set families. Consider a finite universe S. Sets X, $Y \subseteq S$ are *intersecting* if $X \cap Y$, $X - Y$ and $Y - X$ are all nonempty. If in addition $X \cup Y \neq S$ then X and Y are *crossing*. Let $\mathcal{F}$ be a family of subsets of S. $\mathcal{F}$ is a *ring family* if it is closed under intersection and union. $\mathcal{F}$ is an *intersecting* (*crossing*) *family* if $X \cap Y$, $X \cup Y \in \mathcal{F}$ whenever X, $Y \in \mathcal{F}$ and X and Y are intersecting (crossing) [Fu90, GLS]. These three families commonly arise as the minimizers of a submodular function (see Section 7). For example the minimum s,t-cuts of a network form a ring family; the minimum (unrestricted) cuts of a network form a crossing family. We generalize Jordan's result by showing that any intersecting family has a centroid.

We apply the centroid theorem to construct compact representations of set families. Let $n = |S|$. It is well-known that a ring family over S can be represented by a digraph of size $O(n^2)$. An intersecting or crossing family can be represented by n or $2n$ ring families respectively, giving size $O(n^3)$. We present a representation of size $O(n^2)$ for intersecting or crossing families. For specific families the improvement is similar. For example it is well-known that the minimum s,t-cuts of a network of n vertices and m edges can be represented in $O(m)$ space [PQ]. Thus the minimum cuts of a network can be represented in $O(nm)$ space. Our representation uses $O(m \log(n^2/m))$ space.

We apply the representation to improve several submodular flow algorithms. A common bottleneck in these algorithms is an operation on intersecting or crossing families that can be done efficiently using our representation. For example consider the minimum-cost feedback arc set problem. Although NP-complete in general, the problem on planar graphs is equivalent to a submodular flow problem. We solve the minimum-cost planar feedback arc set problem in time $O(n^3)$, improving the bound of $O(n^4)$ due to Frank [F81].

We now describe the results and the organization of the paper in detail. The first part of the paper treats intersecting and crossing families in general. Section 2 gives a broadened definition of centroid and proves our centroid theorem. It also justifies the broadened definition as a direct generalization of ordinary centroids. Section 3 defines the representation and analyzes it for general families. Section 4 gives algorithms to construct the representation, for general families. Section 5 improves the analysis for families derived from graphs. Section 6 shows how the representation can be used to efficiently answer certain queries on intersecting and crossing families. These queries arise in submodular flow algorithms.

The second part of the paper gives examples for the centroid theorem and the representation.

Section 7 describes how in general, submodular functions give rise to the examples discussed. Now we describe the next three sections, comparing the results to previous ones.

Section 8 discusses the family of minimum cuts of a network. As already mentioned the naive representation uses $O(nm)$ space. It can be constructed in time $O(nm \log(n^2/m))$ using the mincut algorithm of Hao and Orlin [HO]. Our representation using $O(m \log(n^2/m))$ space can be constructed in the same time. On a digraph (i.e., a unit capacity network) with edge-connectivity λ the time to construct the representation of all connectivity cuts is $O(\lambda m \log(n^2/m))$.

We mention some related results. For undirected graphs or networks the cactus representation of all minimum cuts, due to Dinits, Karzanov and Lomosonov [DKL], uses $O(m)$ space. It can be constructed in the same two time bounds given above that we achieve for digraphs [Ga91b]. For digraphs [Ga91b] gives a representation of the "minimal mincuts." It does not represent all the mincuts.

Section 9 discusses the family of minimum vertex cuts of a digraph or network. For real-valued vertex capacities our representation has size $O(nm \log(n^2/m))$, versus $O(n^2 m)$ for the naive representation. For digraphs with vertex connectivity κ the representation has size $O(\kappa m \log(n^2/m))$ and can be constructed in time $O(\kappa \min\{\kappa, \sqrt{n}\} nm)$. A previous representation uses space $O((\kappa^2+n)m)$ and can be constructed in time $O((\kappa^2 + n)\min\{\kappa, \sqrt{n}\}m)$ [Ev]. Our submodular flow application (the k-vertex-reachability problem in [Ga93b]) needs only one of the κ families represented, so our time and space bounds decrease by a factor of κ; however the only change in the previous representation (in time and space) is that the factor $(\kappa^2 + n)$ decreases to n.

Kanevsky gives a representation for the minimum vertex separators of an undirected graph [Kan]. It has size $O(\kappa^2 n)$, and can be used to enumerate all minimum separators. Kanevsky shows the number of such separators is $O(2^\kappa n^2/\kappa)$. Our representation represents the minimum separators plus the disconnected pieces.

Section 10 discusses families related to count matroids. These families have applications in the study of graph rigidity and scene analysis. For example when graphs model bar-and-joint frameworks in the plane, rigidity properties are often revealed by the rigidity matroid [R]. The rigidity matroid is a special case of a count matroid. The family of rigid subgraphs of a minimally rigid graph is intersecting. Our representation has size $O(n \log n)$ and is constructed in time $O(n^2)$. A representation using $O(n^2)$ space is proposed in [N] but nonrigid sets can appear in the representation [Ga91b].

The last part of the paper discusses applications to submodular flow algorithms. Section 11 briefly sketches Frank's algorithm for 0-1 submodular network flow [F82]. It shows how the queries

of Section 6 arise in the algorithm.

Section 12 discusses the minimum-cost dijoin problem. This problem is to make a digraph strongly connected by contracting a set of edges of minimum cost. Frank solves the problem in $O(n^3m)$ time using $O(m)$ space, or $O(n^2\bar{M}(n))$ time using $O(n^3)$ space [F81]. Here $\bar{M}(n)$ is the time to multiply two $n \times n$ matrices. Seemingly more complicated algorithms are given in [Kar], [Lu]. Using our representation Frank's algorithm runs in $O(n^2m)$ time using $O(m)$ space, or alternatively $O(nM(n))$ time using $O(n^2)$ space. The planar feedback arc set problem amounts to the minimum-cost dijoin problem on the planar dual graph. Thus as mentioned above we solve the planar feedback problem in time $O(n^3)$ and space $O(n)$.

Section 13 presents similar improvements for versions of another submodular flow problem: orient the edges of an undirected graph to make it k-edge connected. We improve the algorithm of [F82] by a similar factor of n.

The representation applies to achieve similar improvements in other submodular flow problems. Two applications (k-edge-reachability orientation and minimum-cost k-vertex-reachability) are discussed in [Ga93b]. We also use the representation to get efficient submodular flow algorithms that work by cost-scaling [Ga93a].

This section concludes with some terminology. $\mathbf{R}$ is the set of real numbers, $\mathbf{R}_+$ the set of nonnegative reals, $\mathbf{Z}$ the set of integers. Let S and T be sets. We use containment $S \subseteq T$ and proper containment $S \subset T$. S and T *meet* if $S \cap T \neq \emptyset$. If e is an element we often abbreviate the singleton set $\{e\}$ to e. Thus for functions f defined on sets, $f(v)$ means $f(\{v\})$. Similarly $S - e$ denotes $S - \{e\}$ and $S + e$ denotes $S \cup \{e\}$. In set expressions $\cap$ has higher precedence than $\cup$, e.g., $A \cup B \cap C$ denotes $A \cup (B \cap C)$. For a function $f : S \to \mathbf{R}$ and any set $X \subseteq S$, $f(X)$ denotes $\sum\{f(x) \mid x \in X\}$.

A *digraph* $G = (V, E)$ has no parallel edges. The graphs we work with are mainly directed; for the problems we consider (such as edge and vertex connectivity) an undirected graph can be modelled by the corresponding digraph where each edge has both orientations. For $S \subseteq V$, $\rho(S)$ denotes the number of edges directed from $V - S$ to S. $\delta(S)$ denotes $\rho(V - S)$. A *network* is a digraph with parallel edges or more generally, a digraph with an edge-capacity function $c : E \to \mathbf{R}_+ + \infty$. (In Section 9 networks have vertex capacities.) $\rho_c(S)$ denotes the total capacity of edges directed from $V - S$ to S. We abbreviate ρ_c to ρ if the capacity function c is clear. The same applies to δ.

2. Centroids of intersecting families

This section introduces some basic notions including that of a centroid. It proves that any

intersecting family has a centroid. It gives other basic properties of centroids.

A *triangle* consists of three sets that are pairwise intersecting but have no common element. Let $\mathcal{F}$ be a family of subsets of S. An *$\mathcal{F}$-set* is a set in $\mathcal{F}$. An *$\mathcal{F}$-triangle* is a triangle of three $\mathcal{F}$-sets. $\mathcal{F}$ is *triangle-free* if no $\mathcal{F}$-triangle exists.

To illustrate consider the family of subtrees of a tree. We can identify a subtree by its vertices or its edges. In both cases the family is intersecting. When subtrees are edge sets there may be triangles. For instance in a star of three edges, the three subtrees of two edges each form a triangle. When subtrees are vertex sets the family is triangle-free. (To prove this suppose A, B, C is a triangle. Root the tree at a vertex $r \in A \cap B$. Let s be the root of C. Since A and C intersect, A contains a path from r to s. Thus $s \in A$. Similarly $s \in B$. Thus $s \in A \cap B \cap C$, a contradiction.)

The following properties of triangles are useful.

Lemma 2.1. Let $\mathcal{F}$ be an intersecting family.

(i) For an $\mathcal{F}$-triangle A_1, A_2, A_3, $\mathcal{F}$ contains any union of one or more of the sets $A_i \cap A_j$, $1 \leq i \leq j \leq 3$.

(ii) $\mathcal{F}$ is triangle-free if any two nonempty disjoint $\mathcal{F}$-sets A and B have $A \cup B \notin \mathcal{F}$.

(iii) $\mathcal{F}$ is triangle-free iff any two intersecting $\mathcal{F}$-sets A and B have $A \oplus B \notin \mathcal{F}$.

Proof. In this proof it is convenient to denote set intersection by juxtaposition, i.e, AB is an abbreviation for $A \cap B$.

(i) We verify the three most interesting cases and leave the others to the reader. $A_1A_2 \cup A_1A_3$ is in $\mathcal{F}$ since it equals $A_1(A_2 \cup A_3)$. This fact (applied twice) implies that $A_1A_2 \cup A_1A_3 \cup A_2A_3 \in \mathcal{F}$. Finally $A_1 \cup A_2A_3$ is in $\mathcal{F}$ since it equals $(A_1 \cup A_2)(A_1 \cup A_3)$.

(ii) An $\mathcal{F}$-triangle A, B, C gives $\mathcal{F}$-sets AB and AC that violate the condition of (ii), by (i).

(iii) Suppose $\mathcal{F}$ is triangle-free. If A and B are intersecting $\mathcal{F}$-sets with $A \oplus B \in \mathcal{F}$ then A, B, $A \oplus B$ is an $\mathcal{F}$-triangle, a contradiction.

For the converse suppose A, B, C is an $\mathcal{F}$-triangle. Then so is $AB \cup AC$, $BA \cup BC$, $CA \cup CB$, by (i). Thus sets $AB \cup AC$ and $BA \cup BC$ violate the condition. ∎

A family satisfying condition (ii) is obviously not a ring family. In general a triangle-free family is not a ring family if it has three pairwise-disjoint sets.

Now we present the centroid theorem. Let $\mathcal{F}$ be an intersecting family of sets over S. Let $n = |S|$. Let $\mathcal{F}^+$ be the family of all $\mathcal{F}$-sets with $> n/2$ elements. Note that any two $\mathcal{F}^+$-sets meet.

(This is the only property of $\mathcal{F}^+$ used in the following definitions and theorem, a fact that we use below.)

A *centroid* e, A, B consists of an element e and sets A, B such that any $\mathcal{F}^+$-set contains e or forms an $\mathcal{F}$-triangle with A and B. If the first alternative always holds, i.e., e is in every $\mathcal{F}^+$-set, then e is an *element centroid.* In this case A and B are irrelevant. e, A, B is a *triangle centroid* if e is not an element centroid. In a triangle centroid A and B are $\mathcal{F}$-sets.

For any $e \in S$ let $M^+(e)$ be the maximal $\mathcal{F}^+$-set not containing e, if it exists. Since $\mathcal{F}$ is intersecting, $M^+(e)$ is unique if it exists. $M^+(e)$ exists iff e is not an element centroid.

Theorem 2.1. Any intersecting family has a centroid.

Proof. Let A be a minimal $\mathcal{F}$-set that meets every $\mathcal{F}^+$-set. (A exists since any two $\mathcal{F}^+$-sets meet.) Take any $e \in A$. If $M^+(e)$ does not exist then e is an element centroid. Otherwise $M^+(e) \cap A$ is an $\mathcal{F}$-set, properly contained in A. Thus some $\mathcal{F}^+$-set B is disjoint from $M^+(e) \cap A$. We prove e, A, B is a triangle centroid by showing that any $\mathcal{F}^+$-set D not containing e forms a triangle with A and B. The definition of A shows it meets both B and D. B meets D since both are $\mathcal{F}^+$-sets. Finally no element is common to A, B and D since $D \cap A \subseteq M^+(e) \cap A$. ∎

A special case of the theorem is that any triangle-free family has an element centroid. Neither the theorem nor Lemma 2.1(i) holds for arbitrary crossing families.

Note that a triangle centroid e, A, B has $e \in A \cap B$. This follows since $M^+(e) \cup A \cap B$ is an $\mathcal{F}^+$-set (by Lemma 2.1(i)) properly containing $M^+(e)$.

Section 5 uses a slightly more general centroid theorem. Consider a weight function $w : S \to \mathbf{R}_+$. Define $\mathcal{F}^+$ as the family of all $\mathcal{F}$-sets X with $w(X) > w(S)/2$. The definition of centroid is unchanged. Since any two $\mathcal{F}^+$-sets meet, Theorem 2.1 still holds: An intersecting family with an arbitrary nonnegative weight function has a centroid. In Sections 3–4 the main development is stated without weight functions. Remarks indicate any minor changes needed to extend the results to weight functions.

To illustrate the centroid theorem consider again the family of subtrees of a tree T. When subtrees are vertex sets T has an element centroid – this is an ordinary centroid c of T. Now we show that when subtrees are edge sets, centroids still correspond to ordinary centroids of T. This will justify our new notion of centroid as a proper generalization of the ordinary centroid of a tree. First note that there are trees that do not have an element centroid, e.g., 3 paths with a common endpoint. (Note also that the family of subtrees for this graph is not a ring family.)

To do this consider a tree T with n vertices, and ordinary centroid vertex c. T has $n-1$ edges, so a subtree of $\mathcal{F}^+$ has $\geq \lceil n/2 \rceil$ edges. In this argument for a set of vertices S let $\delta(S)$ denote the set of edges with precisely one vertex in S.

First we construct the centroid given by Theorem 2.1. The edges of $\delta(c)$ correspond naturally to trees of $T-c$, so we identify such trees by their corresponding edge. Let e be the edge joining c to a largest tree of $T-c$. If $T-c$ has a tree of precisely $\lfloor n/2 \rfloor$ vertices, then e is an element centroid (deleting e leaves subtrees of $\leq \lceil n/2 \rceil - 1$ edges). The remaining case is when every tree of $T-c$ has $< \lfloor n/2 \rfloor$ vertices. Take set A to be a maximal set of edges of $\delta(c)$ such that $e \in A$ and the trees of A have a total of $\geq \lfloor n/2 \rfloor$ vertices. Clearly $A - e \neq \emptyset$. The choice of e ensures that $A-e$ has $\leq \lfloor n/2 \rfloor - 1$ vertices. (If $f \in A-e$ then the trees of $A-f$ have $< \lfloor n/2 \rfloor$ vertices, whence the trees of $A-e$ have $< \lfloor n/2 \rfloor$ vertices.) B consists of e plus $\delta(c) - A$. ($B - e \neq \emptyset$ since the number of vertices in trees of A is $\leq 2(\lfloor n/2 \rfloor - 1) \leq n-2$.) The number of edges in trees of $B-e$ is $\leq (n-1) - \lfloor n/2 \rfloor = \lceil n/2 \rceil - 1$; this bound also holds for $A-e$. Now it is easy to check that any subtree of $\geq \lceil n/2 \rceil$ edges not containing e forms a triangle with A and B.

Now we show that any centroid arises essentially in this way. More precisely any centroid e, A, B has $e \in \delta(c)$. Furthermore in a triangle centroid e, A, B, $A \cap B \cap \delta(c) = \{e\}$ and edges not in $\delta(c)$ can be deleted from A or B.

To prove $e \in \delta(c)$, let $e = xy$, where x is in the larger tree of $T-e$. We show that x is an ordinary centroid, i.e., no tree of $T-x$ has $\geq \lceil (n+1)/2 \rceil$ vertices. Suppose S is a tree of $T-x$ with $\geq \lceil (n+1)/2 \rceil$ vertices. Let f be the edge joining S to x ($f \neq e$ by the choice of x). The tree $S+f$ of $T-e$ has $\geq \lceil (n+1)/2 \rceil$ edges. Thus e is not an element centroid. If e, A, B is a triangle centroid then $e \in A \cap B$, as noted above. Since A and B are both subtrees intersecting $S+f$ and containing e, they both contain f. But now A, B, $S+f$ is not a triangle since f is in all three sets.

Next consider a triangle centroid e, A, B. A subtree S of $\mathcal{F}^+$ contains c, since otherwise it has $\leq \lfloor n/2 \rfloor$ vertices, whence $\leq \lfloor n/2 \rfloor - 1$ edges. If S forms a triangle with A and B then it forms a triangle with $A \cap \delta(c)$ and $B \cap \delta(c)$. Thus $e, A \cap \delta(c), B \cap \delta(c)$ is a triangle centroid. If e is not an element centroid then the larger subtree of $T-e$ forms a triangle with A and B only if $A \cap B \cap \delta(c) = \{e\}$. This completes the proof showing that in trees, our centroids are essentially ordinary centroids.

Note that a plausible strengthening of our description of centroids is false: Simple examples show that in a triangle centroid, e and A, B need not partition the edges of $\delta(c)$.

Element centroids have these two obvious properties used in Section 3 to construct a representation: For any element e, the collection of all $\mathcal{F}$-sets containing e is a ring family. Any $\mathcal{F}$-set not

containing e is contained in a maximal $\mathcal{F}$-set of $S - e$. We now give analogs of these properties for triangle centroids.

Lemma 2.2. For an $\mathcal{F}$-triangle $\Delta = A, B, C$ define the family of sets

$$\mathcal{F}_\Delta = \{X \mid X \subseteq C,\ X \cup A \cap B \in \mathcal{F}\}.$$

(i) $\mathcal{F}_\Delta$ is a ring family contained in $\mathcal{F} \cup \{\emptyset\}$.

(ii) $\mathcal{F}_\Delta$ contains any $\mathcal{F}$-set $D \subseteq C$ that forms a triangle with A and B.

(iii) Let B' be the maximal $\mathcal{F}$-set containing $B \cap C$ and disjoint from A. The $\mathcal{F}$-sets contained in C but not forming a triangle with A and $B \cup B'$ are precisely the $\mathcal{F}$-sets contained in $B' \cap C$ or in a maximal $\mathcal{F}$-set of $C - B'$.

Proof. (i) $\mathcal{F}_\Delta$ is a ring family since $\mathcal{F}$ is intersecting. $\mathcal{F}_\Delta \subseteq \mathcal{F} \cup \{\emptyset\}$ since a set X in $\mathcal{F}_\Delta$ equals $(X \cup A \cap B) \cap C$.

(ii) follows from Lemma 2.1(i).

(iii) Let D be an $\mathcal{F}$-set contained in C. If D meets both A and $B \cup B'$ it forms a triangle with A and $B \cup B'$. If D meets B' but not A it is contained in B'. If $D \subseteq C - B'$ then it is contained in a maximal $\mathcal{F}$-set of $C - B'$. ∎

Section 3 applies this result to the triangle $A, B, C = M^+(e)$ where e, A, B is a triangle centroid. Observe that in this case $B' \subseteq M^+(e)$. This follows from the maximality of $M^+(e)$.

3. The tree-of-posets representation

This section uses the Centroid Theorem to obtain a compact representation for intersecting families. It also extends the representation to crossing families.

The constructions use the following notion: Fix an intersecting family $\mathcal{F}$ over S. For any set $X \subseteq S$, define $\mathcal{F}(X)$ to be the family of all $\mathcal{F}$-sets contained in X. Clearly $\mathcal{F}(X)$ is an intersecting family.

We begin by defining a tree that forms the core of our representation. For any $X \subseteq S$ the *centroid tree* $\mathcal{C}(X)$ is a labelled ordered tree. Suppose $\mathcal{F}(X)$ has a centroid e, A, B. The root of $\mathcal{C}(X)$ is labelled by e. The subtrees of the root are the trees $\mathcal{C}(Y)$ for Y ranging over the following sets: If e is an element centroid then Y ranges over the maximal $\mathcal{F}$-sets of $X - e$. If e, A, B is a triangle centroid then define the set B' of Lemma 2.2(iii) for triangle $A, B, C = M^+(e)$; Y ranges over B' and the maximal $\mathcal{F}$-sets of $X - B' - e$. Furthermore $\mathcal{C}(B')$ is the first subtree of the root. Define the centroid tree $\mathcal{C}(\mathcal{F})$ as $\mathcal{C}(S)$.

The ordering of children in a centroid tree is used in just a simple way: As noted in Section 2, $B' \subseteq M^+(e)$. This allows $M^+(e)$ to be easily recomputed (see Section 5).

We use the following terminology for centroid trees. Each node x of $\mathcal{C}(\mathcal{F})$ corresponds to a set $X \subseteq S$; S_x denotes this set X. Thus the subtree rooted at x is $\mathcal{C}(S_x)$. S_x is an $\mathcal{F}$-set except possibly for $S_x = S$. For each $v \in S$, $\nu(v)$ is the deepest node x of $\mathcal{C}(\mathcal{F})$ with $v \in S_x$. Thus $v \in S_y$ precisely when y is an ancestor of $\nu(v)$.

It is clear that $\mathcal{C}(\mathcal{F})$ has $\leq n$ nodes (since all node labels are distinct elements).

Suppose x and y are nodes of $\mathcal{C}(\mathcal{F})$, with y a child of x. Then

$$|S_y| \leq |S_x|/2. \tag{1}$$

To prove this let node x have label e. If e is an element centroid then (1) follows from the definitions. If e corresponds to a triangle centroid then (1) follows from the definitions and Lemma 2.2(*iii*).

An obvious corollary to (1) is that tree $\mathcal{C}(\mathcal{F})$ has height $\leq \log n$.

Now we extend the centroid tree to a representation. We make use of the well-known fact that any ring family $\mathcal{F}$ can be represented by a poset P. Recall that in this representation, the $\mathcal{F}$-sets correspond one-to-one with the ideals of P [e.g., GLS, p.314]. Equivalently $\mathcal{F}$ is represented by a directed acyclic graph (dag) D; each node of D represents one or more elements of S; the $\mathcal{F}$-sets correspond to the closed sets of D. (A set is *closed* if it contains all successors of its nodes.) We use the notation that an element $e \in S$ corresponds to node $[e]$ in P or D ($[e]$ does not exist if e is not in any $\mathcal{F}$-set). Thus the smallest $\mathcal{F}$-set containing e consists of $[e]$ and its successors.

For an intersecting family $\mathcal{F}$, the collection of all $\mathcal{F}$-sets containing e is a ring family. So this family can be represented by a poset, which we denote P_e.

Now consider an arbitrary intersecting family $\mathcal{F}$. We enlarge the labels of $\mathcal{C}(X)$ to get a labelled tree $\mathcal{T}(X)$ representing the family $\mathcal{F}(X)$. Each node of $\mathcal{T}(X)$ is labelled with an element of X and one or two posets. Let x be a node with label e in $\mathcal{C}(X)$. If e is an element centroid of S_x then in $\mathcal{T}(X)$, x is labelled by element e and poset P_e for family $\mathcal{F}(X)$. If e, A, B is a triangle centroid of S_x then let Δ be the triangle $A, B, M^+(e)$ and let P_Δ be the poset representing the ring family $\mathcal{F}_\Delta$ of Lemma 2.2 (on family $\mathcal{F}(X)$). Then x is labelled by element e and posets P_e, P_Δ.

For an intersecting family $\mathcal{F}$ over S the *tree-of-posets representation*, denoted $\mathcal{T}(\mathcal{F})$, is $\mathcal{T}(S)$. $|\mathcal{T}(\mathcal{F})|$ denotes the total size of $\mathcal{T}(\mathcal{F})$, including all its posets. (The size of a posets counts the number of nodes and edges.) In the following theorem assume for convenience that $\emptyset \notin \mathcal{F}$.

Theorem 3.1. For any intersecting family $\mathcal{F}$ there is a one-to-many correspondence between $\mathcal{F}$ and the nonempty closed sets of posets of $\mathcal{T}(\mathcal{F})$. The correspondence is one-to-one if $\mathcal{F}$ is triangle-free. $|\mathcal{T}(\mathcal{F})| = O(n^2)$.

Proof. For a triangle-free family the correspondence is clearly one-to-one. For a general family Lemma 2.2(*iii*) ensures that every $\mathcal{F}$-set is represented at least once. (An $\mathcal{F}$-set can be represented in more than one poset because of P_Δ posets. This redundancy is not a problem for the applications we consider.)

To prove the size bound observe that for any i, all sets S_x with

$$n/2^{i+1} < |S_x| \leq n/2^i$$

are disjoint. (This follows from (1) and the fact that the children y of a node x have disjoint sets S_y.) Thus the total size of all posets of $\mathcal{T}(\mathcal{F})$ is $\leq \sum_0^\infty 2^{i+1}(n/2^i)^2 = O(n^2)$. ∎

When S has a weight function w we define $\mathcal{C}(\mathcal{F}, w)$ and $\mathcal{T}(\mathcal{F}, w)$ exactly as in the above discussion but using weighted centroids. The part of Theorem 3.1 on correspondences holds for representations $\mathcal{T}(\mathcal{F}, w)$, but not the bound on $|\mathcal{T}(\mathcal{F})|$ (or the bound on the height of $\mathcal{C}(\mathcal{F})$).

Note that P_Δ can be constructed from P_e: P_Δ is the subgraph of P_e induced by the nodes comprising C. In proof recall that $\mathcal{F}_\Delta$ consists of the sets $X \subseteq C$ such that $X \cup A \cap B \in \mathcal{F}$. These sets are precisely the sets $Y \subseteq C$ such that $Y \cup A' \in \mathcal{F}$ for some set A' disjoint from C and containing e. (Recall from Section 2 that $e \in A \cap B$; also $C \cup A \cap B \in \mathcal{F}$.) The sets Y are precisely the sets represented in the subgraph induced by C.

We shall see that the posets labelling a node of $\mathcal{T}(\mathcal{F})$ can be constructed efficiently from the information in $\mathcal{C}(\mathcal{F})$. This allows centroid trees to be used as a representation in some contexts (see Section 5).

We now distinguish two more versions of the representation. A poset P of $\mathcal{T}(\mathcal{F})$ can be stored in complete, transitively-closed form or as a dag whose transitive closure is P. The notation $\mathcal{T}(\mathcal{F})$ refers to the representation using either dags or posets; $\mathcal{T}^c(\mathcal{F})$, the *transitively-closed representation*, refers to the representation using complete posets. Section 3.4 illustrates how transitively-closed representations can be used to trade space for time.

The transitively-closed representation can be constructed efficiently. Let $M(n)$ be the optimal time to multiply two n by n matrices. Assume $M(n) = \Theta(n^2 t(n))$ for some nondecreasing function $t(n)$. We claim $\mathcal{T}^c(\mathcal{F})$ can be constructed from $\mathcal{T}(\mathcal{F})$ in time $O(M(n))$. Similarly if $M(n) = O(n^2 t(n))$ the time is $O(n^2 t(n))$. It is known that $M(n) = O(n^{2.38})$ [CW], so $\mathcal{T}^c(\mathcal{F})$ can be constructed from $\mathcal{T}(\mathcal{F})$ in time $O(n^{2.38})$.

To prove the claim recall from the proof of Theorem 3.1 that all sets S_x of $\mathcal{C}(\mathcal{F})$ with $n/2^{i+1} < |S_x| \leq n/2^i$ are disjoint. $M(n)$ bounds the time to form a transitive closure. Thus the total time to find the transitive closure of all posets of $\mathcal{T}(\mathcal{F})$ is at most a constant times $\sum_0^\infty 2^{i+1}(n/2^i)^2 t(n) = O(\bar{n}^2 t(\bar{n}))$.

We conclude this section by extending the representation to crossing families. Choose any element s. Define $\mathcal{F}^s$ to be the family of all $\mathcal{F}$-sets not containing s. $\mathcal{F}^s$ is an intersecting family. Define $\mathcal{F}_s$ to be the family of complements of all $\mathcal{F}$-sets containing s. Symmetrically $\mathcal{F}_s$ is an intersecting family. (This follows since if $\mathcal{F}$ is a crossing family, the complements of all $\mathcal{F}$-sets form another crossing family.) $\mathcal{F}$ consists of the sets of $\mathcal{F}^s$ and the complements of the sets of $\mathcal{F}_s$. We represent $\mathcal{F}^s$ by $\mathcal{T}(\mathcal{F}^s)$ and $\mathcal{F}_s$ by $\mathcal{T}(\mathcal{F}_s)$. Thus the $\mathcal{F}$-sets correspond to the ideals of the posets in $\mathcal{T}(\mathcal{F}^s)$ and the "dual ideals" (i.e., complements of ideals [Fu90]) in $\mathcal{T}(\mathcal{F}_s)$.

4. Constructing the representation

This section gives algorithms to construct the representation. We assume an oracle that supplies the poset P_e for every element e. More precisely every oracle call returns a new element e and the corresponding poset P_e. (This models specific oracles where the sequence of elements e and posets P_e is unpredictable, as in Section 5.) Other oracles may be more conveniently implemented for certain set families, perhaps requiring different algorithms.

We state resource bounds using the following notation. Let each poset P_e have at most m nodes and edges ($m \leq n^2$). Let τ and σ denote the time and space, respectively, used by the oracle in the course of returning all n of the posets P_e, $e \in S$. (A poset P_e need not be retained after it is returned by the oracle, unless the algorithm needs it for later computations.)

We begin by giving two algorithms to find a centroid. The first algorithm is for triangle-free families and the second is for general families. At the end of this section we indicate the advantage of the first algorithm.

For elements v, w let $M(v,w)$ denote the maximal $\mathcal{F}$-set containing v but not w, if such a set exists; otherwise $M(v,w) = \emptyset$. Observe that w is an element centroid iff $|M(v,w)| \leq n/2$ for every v.

First consider a triangle-free family $\mathcal{F}$. We use the following fact to limit the search for a centroid.

Lemma 4.1. Let $\mathcal{F}$ be an arbitrary intersecting family (not necessarily triangle-free). For elements v, w, if $|M(v,w)| \leq n/2$ and v is an element centroid then so is w.

Proof. If w is not a centroid then $M^+(w)$ exists. The hypothesis implies $v \notin M^+(w)$. Thus v is not a centroid. (This argument holds even if $M(v,w) = \emptyset$.) ∎

Now we present the algorithm to find a centroid. It works by examining posets P_e in turn, maintaining a set C of candidate centroids. More precisely these invariants hold throughout the algorithm: C contains an element centroid. For any previously examined poset P_e and any $x \in C$, $|M(e,x)| \leq n/2$.

The algorithm initializes C to S. It repeats the following until every poset P_e, $e \in S$, has been examined. Call the oracle to return a new element e and poset P_e. (The sequence of elements e is determined by the oracle.) Search P_e to find some $c \in C$ with $M(e,c)$ maximal subject to the constraints that $|M(e,c)| \leq n/2$. In other words, $\leq n/2$ elements of S are in nodes of P_e that do not precede $[c]$, and no element $x \in C$ with $[x]$ strictly preceding $[c]$ has this property. Assign $C \leftarrow C \cap [c]$ and continue to the next poset. After all posets have been examined, any element of C is a centroid.

In this algorithm note that in a poset P_e an element x may have $[x]$ undefined (i.e., if no $\mathcal{F}$-set contains both e and x). In this case the algorithm interprets $[x]$ to be the set of all elements with $[x]$ undefined; it assumes that $[x]$ precedes all nodes of P_e.

To show the algorithm is correct we need only verify that it maintains the invariants. Consider any iteration. The element c chosen by the algorithm exists since C contains a centroid. Let x be an element deleted from C. Thus $x \notin [c]$. If $x \in M(e,c)$ then $M(x,c) = M(e,c)$; now Lemma 4.1 implies that x can be discarded. The second possibility for x is that $x \notin [c] \cup M(e,c)$. (In this case $[c]$ is defined.) Since $[c] \cup M(e,c) \in \mathcal{F}$, and $|[c] \cup M(e,c)| > n/2$ (by the choice of c), x is not a centroid.

Now we show the time for this algorithm is $O(\tau + nm)$. The oracle constructs all posets P_e. We need only show that the remaining work is $O(nm)$.

To implement the search of an iteration, examine nodes x of P_e in topological order (i.e., examine all predecessors of x before x). If x contains an element of C, compute $|M(e,x)|$ by marking all predecessors of x. If $|M(e,x)| \leq n/2$ then take x as c and end the search. Otherwise continue by examining the next node x.

One iteration of this algorithm spends $O(m)$ time to delete an element (contained in) x from C, and $O(m)$ additional time to find c. Since $\leq n$ elements are deleted from C and there are n iterations, this gives $O(nm)$ time total.

It is easy to check that the space for the algorithm is $O(\sigma + m)$.

We turn to finding a centroid for a general intersecting family. The basic algorithm follows the proof of Theorem 2.1, maintaining an $\mathcal{F}$-set A that meets every $\mathcal{F}^+$-set. It works as follows.

Initialize A to an arbitrarily chosen $\mathcal{F}^+$-set (possibly S). Then repeat the following steps until a centroid is found. Choose any $e \in A$. If $M^+(e)$ does not exist then e is an element centroid. Otherwise assign $A' \leftarrow M^+(e) \cap A$. Set B to the maximal $\mathcal{F}^+$-set that is disjoint from A'. If such a B exists then e, A, B is a triangle centroid. Otherwise assign $A \leftarrow A'$ and continue with the next iteration.

The proof of Theorem 2.1 justifies this algorithm. (Note that set A is maintained as an $\mathcal{F}$-set that meets every $\mathcal{F}^+$-set.)

We now implement the algorithm using an oracle for P_e. The two main tasks are to find sets $M^+(e)$ and B (or determine they do not exist). To achieve the desired time bound we will choose a specific element e; however similar time bounds are easily achieved without this choice. Assume that at the start of an iteration element e and poset P_e are known.

To find $M^+(e)$, find an element $f \in A - e$ such that $|M(f,e)| > n/2$, and assign $M^+(e) \leftarrow M(f,e)$. The set $M(f,e)$ is found using the poset P_f, which is supplied by the oracle. (We repeatedly call the oracle until it returns P_f for an element $f \in A$. We discard any poset P_g that is returned for a $g \notin A$.) If no such element f exists then e is an element centroid. (This is justified below.)

Next we wish to find set B, the maximal $\mathcal{F}^+$-set disjoint from A'. If B exists then it contains e (since $e \notin B$ implies $B \subseteq M^+(e)$, whence $B \cap A \subseteq A'$, a contradiction). Thus B can be found by examining the poset P_e.

It remains only to discuss how e is chosen. In the first iteration, e can be taken as the first element in A returned by the oracle. For any subsequent iteration, suppose the previous iteration found element $f \in A - e$ with $M^+(e) = M(f,e)$. Choose f as the next value of e. (This is valid since $f \in A$ at the start of the iteration.)

To prove this algorithm is correct we must check that when it returns an element centroid e, the choice is valid. Since A meets every $\mathcal{F}^+$-set, if $M^+(e)$ exists then it contains some $f \in A - e$. Thus it suffices to show this invariant: At the start of any iteration after the first, e is the only element of A whose poset has been returned by the oracle. (The invariant implies the algorithm finds set $M^+(e)$ if it exists, and hence correctly identifies element centroids.) To prove the invariant suppose P_f is constructed in computing $M^+(e)$. If $|M(f,e)| \leq n/2$ then $f \notin M^+(e)$. Thus $f \notin A'$ and f is never again in A. If $|M(f,e)| > n/2$ and element e does not give a triangle centroid then the next iteration chooses f as e.

The time for this algorithm is the time used by the oracle, plus time to examine each poset. It is easy to see that the time spent on each poset is linear in its size. This implies total time $O(\tau + nm)$. We have proved the following result.

Lemma 4.2. For any intersecting family with an oracle for P_e, a centroid can be found in time $O(\tau + nm)$ and space $O(\sigma + m)$. ∎

Now we show how to construct the representation $\mathcal{T}(\mathcal{F})$. Consider first a triangle-free family. Begin by finding an element centroid v. Construct the poset P_v and associate it with the root r of $\mathcal{C}$. Find the distinct sets $M(u,v)$; recurse on each such set, making its tree a subtree of r. To find sets $M(u,v)$ repeat the following: Let u be an element whose set has not been found (if no such element exists then stop). $M(u,v)$ is the maximal closed set of P_u not containing v, if such exists.

This procedure is easily implemented using an oracle for P_e. The time to process the root r is the same as Lemma 4.2.

The algorithm for general intersecting families is similar: As noted in Section 3, for a triangle centroid P_Δ is easily computed from P_e. To construct the children of the root, note that the first child (the set B' of Lemma 2.2(*iii*)) can be constructed from the output of the centroid algorithm, $B' = B \cap M^+(e)$. The other children are the maximal $\mathcal{F}$-sets disjoint from $B' + e$. They are found as in the element centroid case.

We now estimate the time for the construction. As in Section 2 the bound is oriented toward general families with dense posets (better bounds are in Sections 8–10). Considering τ as a function of n we expect $\tau(n) = \Omega(n^3)$, since τ measures the time to construct n posets on n vertices that can have total size $\Theta(n^3)$.

Theorem 4.1. For any intersecting family with an oracle for P_e, $\mathcal{T}(\mathcal{F})$ can be constructed in time $O(\tau(n))$ if $\tau(n) = \Omega(n^3)$.

Proof. The hypothesis and Lemma 4.2 shows that the time spent to construct the root of $\mathcal{T}(\mathcal{F})$ along with its labels, and identify its children, is $O(\tau(n))$. Suppose the algorithm stores every poset P_e, $e \in S$. Then the time at any nonroot node x of $\mathcal{T}(\mathcal{F})$ is easily seen to be proportional to the total size of the posets P_y for all elements $y \in S_x$, i.e., $O(|S_x|^3)$. Thus as in the proof of Theorem 3.1 the total time at nonroot nodes is at most a constant times $\sum_0^\infty 2^{i+1}(n/2^i)^3 = O(n^3)$. ∎

A disadvantage of the approach sketched in this proof is that it needs $O(n^3)$ space to store all posets P_e, $e \in S$. In practice this is not necessary. Usually (e.g., for all examples in this paper) an

oracle is available for not just $\mathcal{F}$, but for all families $\mathcal{F}(X)$. The algorithm uses this oracle to find the posets at each nonroot node of $\mathcal{T}(\mathcal{F})$.

To elaborate suppose such an oracle for posets of $\mathcal{F}(X)$ is available, and on any set X it uses time $\tau(|X|)$. Assume that $\tau(n) = \Theta(n^3 f(n))$ where f is a nondecreasing function. The theorem holds for this case as well: $\mathcal{T}(\mathcal{F})$ can be constructed in time $O(\tau(n))$. The proof is the same as above, with the observation that the total time for all oracles is bounded by $\sum_0^\infty 2^{i+1}(n/2^i)^3 f(n) = O(n^3 f(n))$. The space for the algorithm is the space for $\mathcal{T}(\mathcal{F})$ plus the space used by the oracle.

We close the section by discussing an advantage of the centroid algorithm for triangle-free families. It can be adapted to work with a related oracle, which is the oracle of choice in Section 8. A sequence of calls to the related oracle returns the posets P'_e. Here P'_e is defined as the poset representing all $\mathcal{F}$-sets that contain e but no element f whose poset P'_f was returned in a previous call. (Thus each $\mathcal{F}$-set is represented precisely once in all posets P'_e.)

To adapt the centroid algorithm to the related oracle, consider an execution of the algorithm with the original oracle. Consider an iteration where the oracle returns element e and its poset P_e. If in some previous iteration, the oracle returned an element f and its poset P_f, such that $e \in M(f,c)$ in that iteration, then the current iteration for e can be skipped (since $M(e,c) = M(f,c)$). In the opposite case, let F be the set of all elements whose posets have been previously returned by the oracle. Then $F \cap M(e,x) = \emptyset$ for any $x \in C$. Thus $M(e,x)$ can be computed correctly from the poset P'_e returned by the related oracle.

5. Graph families

This section improves the bounds of Theorem 3.1 and 4.1 for a number of families, including those of Sections 8–10. We state the principle for graphs although it is more general.

For a graph $G = (V, E)$ write $n = |V|$, $m = |E|$. For a set $X \subseteq V$ define

$$\begin{aligned} n_X &= |X|; \\ m_X &= \text{the number of edges with both ends in } X. \end{aligned} \tag{2}$$

Let $\mathcal{F}$ be an intersecting family defined on V. $\mathcal{F}$ has the *sparse poset property* if for any $X \subseteq V$ and $v \in X$, poset P_v for family $\mathcal{F}(X)$ has size $O(n_X + m_X)$. A crossing family $\mathcal{F}$ has the sparse poset property if for any s the intersecting families $\mathcal{F}^s$ and $\mathcal{F}_s$ have the sparse poset property. The following result is for representations without weight functions.

Lemma 5.1. Let $\mathcal{F}$ be an intersecting family with the sparse poset property.

(*i*) $|\mathcal{T}(\mathcal{F})| = O(m \log(n^2/m))$.

(*ii*) Suppose any poset P_v for a family $\mathcal{F}(X)$, $X \subseteq V$ can be constructed in time $O(m_X f(n_X, m_X))$ for f a nondecreasing function of n_X and m_X. Then $\mathcal{T}(\mathcal{F})$ can be constructed in time $O(nmf(n,m))$.

(*iii*) $\mathcal{T}^c(\mathcal{F})$ can be constructed from $\mathcal{T}(\mathcal{F})$ in time $O(nm)$.

Proof. We begin with an observation for all three parts. For any integer i, consider all nodes x of $\mathcal{C}$ such that $n/2^{i+1} < |S_x| \leq n/2^i$. The total number of vertices and edges in graphs induced by these sets S_x is $O(\min\{m, n^2/2^i\})$. To see this recall from the proof of Theorem 3.1 that for a given i, the sets S_x are disjoint. Now the upper bound $O(m)$ follows since a given vertex or edge of G occurs in at most one subgraph induced by a vertex set S_x. The upper bound $O(n^2/2^i)$ follows since the total number of vertices and edges is $\leq 2^{i+1}(n/2^i)^2 = O(n^2/2^i)$.

(*i*) Since $\mathcal{F}$ has the sparse poset property, the above bound also applies to the total size of all posets for the nodes x. Now the size of all posets for values of $i \leq \log(n^2/m)$ is $O(m \log(n^2/m))$ by the first bound. The size for all values of $i \geq \log(n^2/m)$ is $O(m)$ by the second bound.

(*ii*) At the root of $\mathcal{C}$, all posets P_v can be constructed in time $O(nmf(n,m))$. Lemma 4.2 shows the root of $\mathcal{T}(\mathcal{F})$ along with its labels, and the identity of its children can be constructed in the same bound $O(nmf(n,m))$. Thus the total time to construct $\mathcal{T}(\mathcal{F})$ is at most a constant times $\sum_0^\infty (n/2^i)mf(n,m) = O(nmf(n,m))$.

(*iii*) The transitive closure of the posets of the root can be found in time $O(nm)$. As in (*ii*) the time to compute all transitive closures is a constant times $\sum_0^\infty (n/2^i)m = O(nm)$. ∎

For some applications it suffices to store the centroid tree $\mathcal{C}(\mathcal{F})$ rather than $\mathcal{T}(\mathcal{F})$. In addition we store the values $\nu(v)$. Given this information we can efficiently construct the posets of $\mathcal{T}(\mathcal{F})$ labelling a given node x, as follows. Let e be the element labelling x. To construct poset P_e we only need to know e and the vertices of S_x. To construct poset P_Δ first find $C = M^+(e)$ as $M(f,e)$, for any vertex $f \in C$ (choose f as the vertex labelling the first child of x). P_Δ is the subgraph of P_e induced by C.

For graphs we use representations where the weight function is the degree function d, i.e., for any vertex v, $d(v)$ is the total number of edges incident to v (these edges can be directed to or from v). Consider the *centroid tree for the degree function* $\mathcal{C}(\mathcal{F}, d)$. Let $\mathcal{F}$ be an intersecting family with the sparse poset property. Let f be defined as in Lemma 5.1(*ii*).

Lemma 5.2. For any vertex v, the posets for nodes on the path from $\nu(v)$ to the root of $\mathcal{C}(\mathcal{F}, d)$ have total size $O(m)$. They can be constructed in time $O(mf(n, m))$.

Proof. A node x of depth i has total weight $\leq m/2^{i-1}$. This quantity bounds the size of the subgraph induced by S_x. Thus the posets at x have size $O(m/2^i)$. This implies the desired bounds. ∎

6. Intersection queries

This section illustrates how the representation is used for a sample application: For a family $\mathcal{F}$ over S and any $Q \subseteq S$, define $\mu(Q, \mathcal{F})$ as the intersection of all $\mathcal{F}$-sets containing Q. Usually the family $\mathcal{F}$ is clear from context and we write $\mu(Q)$. If no $\mathcal{F}$-set contains Q then $\mu(Q) = S$ by convention. An *intersection query* for $Q \subseteq S$ returns the set $\mu(Q)$. We wish to process a sequence of intersection queries (for a fixed $\mathcal{F}$). Let $q = |Q|$. The submodular flow algorithms of this paper perform intersection queries for $\mathcal{F}$ crossing with $q = 1$; the algorithms of [Ga93b] perform queries for $\mathcal{F}$ intersecting with $q = 1, 2$. We give algorithms for arbitrary q and also note simplifications for these special cases.

Observe that if $\mathcal{F}$ is intersecting, $\mu(Q)$ is S or the smallest $\mathcal{F}$-set containing Q. If $\mathcal{F}$ is crossing, $\mu(Q)$ need not be an $\mathcal{F}$-set.

First consider an intersecting family $\mathcal{F}$. Without loss of generality, enlarge $\mathcal{F}$ so it contains S. $\mathcal{F}$ is still an intersecting family, and now $\mu(Q)$ is always the smallest $\mathcal{F}$-set containing Q.

Looking forward to crossing families we introduce a related function called μ^{-1} (a slight abuse of notation): $\mu^{-1}(Q, \mathcal{F}) = \{u \mid \mu(u) \cap Q \neq \emptyset\}$. As above we abbreviate this to $\mu^{-1}(Q)$.

For a set of elements Q define a poset $P(Q)$ that is a label of $\mathcal{T}(\mathcal{F})$, as follows. Let x be the nearest common ancestor in $\mathcal{C}(\mathcal{F})$ of all nodes $\nu(v)$, $v \in Q$ (e.g., $x = \nu(v)$ for $Q = \{v\}$). $P(Q)$ is one of the posets labelling x: Suppose x is labelled by element e. If e corresponds to a triangle centroid and poset P_Δ contains a node $[v]$ for each $v \in Q$, then $P(Q) = P_\Delta$; otherwise $P(Q) = P_e$. (Recall that $[v]$ exists in all P_e posets along the path from $\nu(v)$ to the root, and possibly in some P_Δ posets.)

Lemma 6.1. Let $\mathcal{F}$ be an intersecting family. $\mu(Q)$ consists of all elements u such that in $P(Q)$, some $[v]$, $v \in Q$, precedes $[u]$. $\mu^{-1}(Q)$ consists of all elements u such that in $P(u)$, $[u]$ precedes some $[v]$, $v \in Q$.

Proof. First consider $\mu(Q)$. Let x be the nearest common ancestor used to define $P(Q)$. Set S_x is an $\mathcal{F}$-set containing Q, so $\mu(Q) \subseteq S_x$. The definition of nearest common ancestor implies that

$\mu(Q)$ is not represented in the poset of any proper descendant of x. Thus $\mu(Q)$ is represented in a poset labelling x.

The set described in the lemma is clearly an $\mathcal{F}$-set containing Q, and it is the smallest such set in a poset labelling x. Thus it is $\mu(Q)$.

The characterization of $\mu^{-1}(Q)$ follows from the characterization of $\mu(u)$. ∎

We estimate the time for all algorithms assuming a complete list of the elements of $\mu(Q)$ or $\mu^{-1}(Q)$ is desired. In this section assume that with $\mathcal{T}(\mathcal{F})$ we store $\nu(v)$ values. Also in this section the parameter m denotes the size of the largest poset in $\mathcal{T}(\mathcal{F})$. (For a crossing family $\mathcal{F}$, m denotes the size of the largest poset in $\mathcal{T}(\mathcal{F}^s)$ or $\mathcal{T}(\mathcal{F}_s)$.)

Lemma 6.2. Let $\mathcal{F}$ be an intersecting family.

(i) Given a transitively-closed representation $\mathcal{T}^c(\mathcal{F})$, $\mu(Q)$ and $\mu^{-1}(Q)$ can be found in time $O(qn)$.

(ii) Given $\mathcal{T}(\mathcal{F})$, $\mu(Q)$ can be found in time $O(m)$. $\mu^{-1}(Q)$ can be found in time $O(|\mathcal{T}(\mathcal{F})|)$.

Proof. The algorithms and time bounds for $\mu(Q)$ follow easily from Lemma 6.1.

To compute $\mu^{-1}(Q)$, visit the posets along the paths from $\nu(v)$ to the root, for each $v \in Q$. For each such poset compute the predecessors of all elements of Q and apply Lemma 6.1.

Now we verify the time bound for $\mu^{-1}(Q)$. For (ii) it is clear that the total time is proportional to the size of all posets visited. For (i) implement the algorithm so that an element u is checked only once, in the poset $P(u)$. Since u is checked in time $O(q)$, the time bound $O(qn)$ follows. ∎

We turn to crossing families $\mathcal{F}$. First we characterize $\mu(Q)$.

Lemma 6.3. Let $\mathcal{F}$ be a crossing family. Fix an element s. For any set $Q \subseteq S$,

$$\mu(Q,\mathcal{F}) = \mu(Q,\mathcal{F}^s) \cap \mu^{-1}(Q,\mathcal{F}_s).$$

Proof. $u \in \mu(Q,\mathcal{F})$ iff u is in every $\mathcal{F}$-set containing Q. u is in every $\mathcal{F}$-set containing Q but not s iff $u \in \mu(Q,\mathcal{F}^s)$. u is in every $\mathcal{F}$-set containing Q and s iff no $\mathcal{F}$-set contains Q and s but not u, iff no complement of an $\mathcal{F}$-set contains u but no member of $Q+s$, iff $\mu(u,\mathcal{F}_s)$ meets Q, iff $u \in \mu^{-1}(Q,\mathcal{F}_s)$. (The lemma holds for cases such as $s \in Q$ or $s \in \mu(Q,\mathcal{F})$ by the conventions on μ. ∎

The previous two lemmas show that an intersection query for a crossing family $\mathcal{F}$ can be answered in these time bounds: Given $\mathcal{T}^c(\mathcal{F}^s)$ and $\mathcal{T}^c(\mathcal{F}_s)$, the time is $O(qn)$. Given $\mathcal{T}(\mathcal{F}^s)$ and $\mathcal{T}(\mathcal{F}_s)$, the time is $O(|\mathcal{T}(\mathcal{F}^s)| + |\mathcal{T}(\mathcal{F}_s)|)$.

We turn to special cases of intersection queries that arise in the submodular flow algorithms mentioned at the start of the section. These cases can be processed efficiently using the centroid tree for the degree function. Define f as in Lemma 5.1(ii) .

Lemma 6.4. Let $\mathcal{F}$ be a family with the sparse poset property.

(i) For $\mathcal{F}$ crossing, given $\mathcal{C}(\mathcal{F}^s, d)$ and $\mathcal{C}(\mathcal{F}_s, d)$ an intersection query can be answered in time $O(qmf(n,m))$ and space $O(m)$.

(ii) For $\mathcal{F}$ intersecting, given $\mathcal{C}(\mathcal{F}, d)$ a sequence of c intersection queries with a common vertex can be answered in time $O(m(f(n,m)+c))$ and space $O(m)$.

Proof. (i) As noted above, to compute $\mu(Q)$ we need only visit the posets (of both centroid trees) on the paths from $\nu(v)$ to the root, for $v \in Q$. Thus Lemma 5.2 implies the result.

(ii) Let v be the common vertex. Before processing the first query, construct and store the posets for nodes on the path from $\nu(v)$ to the root. Lemma 5.2 implies the desired bounds. ∎

The k-vertex-reachability problem of [Ga93b] has the special structure of part (ii): the query sequence breaks up into subsequences with a common vertex.

We conclude by noting bounds that can be achieved without the representation. For intersecting families and queries $\mu(Q)$, the bound $O(m)$ of Lemma 6.2(ii) can be achieved at the expense of storing n posets P_e. The bound $O(qn)$ of Lemma 6.2(i) can be achieved by storing n transitively closed posets. For crossing families and the special case of single-element queries $\mu(u)$, time $O(n)$ can be achieved at the expense of $O(n^2)$ space, using Lemma 6.3.

7. General remarks on examples

The next three sections illustrate our general results by discussing specific intersecting and crossing families. They analyze the representation of each family and give an efficient construction algorithms. The first two examples, minimum edge cuts and minimum vertex cuts, illustrate the two cases for our representation, triangle-free families and general families.

First recall a basic fact that gives rise to intersecting and crossing families. A function $f : \mathcal{F} \to \mathbf{R}$ is *submodular* if $\mathcal{F}$ is a ring family and for all $X, Y \in \mathcal{F}$,

$$f(X) + f(Y) \geq f(X \cup Y) + f(X \cap Y).$$

The minimizers of f form a ring family. The minimizers over $\mathcal{F} - \{\emptyset\}$ ($\mathcal{F} - \{\emptyset, S\}$) form an intersecting (crossing) family. (All these assertions follow easily from the defining inequality.) The intersecting and crossing families in the next three sections all arise in this way.

8. Minimum edge cuts

Consider a network $G = (V, E)$ with edge-capacity function c. A set of vertices X minimizing $\rho(X)$ over $\emptyset \subset X \subset V$ is a *minimum (edge) cut* of G. The *edge connectivity* λ is the minimizing value $\rho(X)$. It is well-known that ρ is submodular on 2^V. Thus the minimum cuts form a crossing family.

We represent the minimum cuts by the intersecting families $\mathcal{F}^s$ and $\mathcal{F}_s$, for an arbitrary vertex s. $\mathcal{F}^s$ consists of the vertex sets $X \subseteq V - s$ having $\rho(X) = \lambda$. $\mathcal{F}_s$ consists of the vertex sets $X \subseteq V - s$ having $\rho(X) = \lambda$ in the graph G with all edges reversed. Thus we represent all minimum cuts using two representations of the form $\mathcal{T}(\mathcal{F}^s)$. (One of these families $\mathcal{F}^s$ can be empty, e.g., if s belongs to every set X having $\rho(X) = \lambda$.)

In the following lemma assume that $\mathcal{F}^s$ nonempty, i.e., some set $X \subseteq V - s$ has $\rho(X) = \lambda$. (This version of the lemma gets used below. However in the opposite case the lemma still holds if λ denotes the minimum value $\rho(X)$ for $\emptyset \subset X \subseteq V - s$.)

Lemma 8.1. $\mathcal{F}^s$ is a triangle-free intersecting family for $\lambda > 0$. It is a ring family for $\lambda = 0$.

Proof. Suppose $\lambda > 0$. For any sets $A, B \subseteq V$ let $d(A, B)$ denote the number of edges that join $A-B$ and $B-A$ (in either direction). Recall the identity $\rho(A)+\rho(B) = \rho(A \cap B)+\rho(A \cup B)+d(A, B)$.

Consider two intersecting sets $A, B \in \mathcal{F}^s$. The identity implies that $A \cup B$, $A \cap B \in \mathcal{F}^s$. Furthermore $d(A, B) = 0$. This implies $\rho(A \oplus B) \geq 2\lambda > \lambda$. Thus $A \oplus B \notin \mathcal{F}^s$. Now Lemma 2.1(*iii*) shows $\mathcal{F}^s$ is triangle-free.

It is easy to see that when $\lambda = 0$ $\mathcal{F}^s$ is a ring family. ∎

The Centroid Theorem now implies that for $\lambda > 0$, $\mathcal{F}^s$ has an element centroid. This can be restated as follows: Consider a network with $\lambda > 0$. Any vertex s has a vertex t, one or both of which are in any minimum cut with more than half the vertices. In general one vertex cannot "cover" all minimum cuts with more than half the vertices (e.g., a directed cycle). Also simple examples show that for $\lambda = 0$, even if G is weakly connected such a cover may require $\Theta(n)$ vertices.

Some submodular flow algorithms use a more general notion of minimum edge cuts, allowing vertex weights (see [F84, Ga93a]). Each vertex v has a real-valued weight $w(v)$ ($w(v)$ can be

positive, negative or zero). A set of vertices X minimizing $\rho(X) + w(X)$ over $\emptyset \subset X \subset V$ is a minimum cut. Define λ and $\mathcal{F}^s$ as above. The new version of Lemma 8.1 states that $\mathcal{F}^s$ is triangle-free as long as $\lambda \neq 0$. The proof is exactly as before. However in the rest of this section we return to ordinary edge cuts.

Lemma 8.1 implies it is a simple matter to represent all minimum cuts of a network having $\lambda = 0$. Specifically the dag representation of $\mathcal{F}^s$ has nodes that are the strong components of the graph G; the edges are those edges of G joining two distinct strong components. So there is no need for our representation $\mathcal{T}(\mathcal{F}^s)$ when $\lambda = 0$. Now it will be convenient to restrict the discussion of $\mathcal{T}(\mathcal{F}^s)$ to networks with $\lambda > 0$.

Our representation uses the representation of Picard and Queyranne for all minimum s,t-cuts of a network [PQ]. This representation is constructed as follows. (For convenience we describe the representation of all sets T containing t but not s and having minimum indegree $\rho(T)$.) Start with the residual graph for a maximum flow from s to t. Contract each strong component. Furthermore contract t and all its predecessors; delete s and all its successors; add an edge from $[t]$ to each source $\neq [t]$. The minimum cuts containing t but not s correspond one-to-one with the sets having no entering edges in this dag. The dag has at most $m + n$ edges. It can be constructed from the residual graph in linear time.

A similar construction applies if we start with the residual graph for a maximum preflow from s to t. To wit, note that a minimum cut containing t but not s is a set containing t but not s or any vertex with positive excess, and having no entering residual capacity. Thus the only change in the above construction is that all vertices that are successors of either s or a vertex with positive excess get deleted.

The following discussion is stated for $\mathcal{F}^s$ but as mentioned, it applies to the representation of all minimum cuts of a network.

Theorem 8.1. For $\mathcal{F}^s$ defined for edge cuts of a network with $\lambda > 0$, $|\mathcal{T}(\mathcal{F}^s)| = O(m \log(n^2/m))$.

Proof. A poset P_v for a family $\mathcal{F}^s(X)$ is constructed as follows. Start with the subnetwork induced by X. Add a new vertex s. In addition for each vertex $x \in X$ add an edge sx whose capacity equals the total capacity of all edges directed to x from a vertex not in X. Poset P_v is the Picard-Queyranne representation of all s,v-cuts. (The poset is empty if a minimum s,v-cut has capacity larger than λ.) P_v has size $O(n_X + m_X)$, so $\mathcal{F}^s$ has the sparse poset property. Now the theorem follows from Lemma 5.1(i) . ∎

Next we discuss algorithms for constructing $\mathcal{T}(\mathcal{F}^s)$. We begin with an algorithm that works on any network with $\lambda > 0$.

The main step in constructing $\mathcal{T}(\mathcal{F}^s)$ is finding a centroid. We do this using the mincut algorithm of Hao and Orlin, which finds a minimum cut in a network in time $O(nm\log(n^2/m))$ [HO]. We first briefly review the mincut algorithm. The algorithm finds, for a given network and distinguished vertex s, a minimum cut containing s. It does this by processing the vertices of $V - s$ in a certain order. We use this order to identify the vertices of $V - s$, so $V - s = \{1, \ldots, n-1\}$. The algorithm computes $n - 1$ maximum value preflows. The ith such preflow has sink vertex i and source vertices $s, 1, \ldots, i-1$ (equivalently these i vertices are contracted into one source vertex). The desired minimum cut is found from the preflow of smallest value. The total time for the algorithm is $O(nm\log(n^2/m))$.

We find a centroid of $\mathcal{F}$ using the centroid algorithm for triangle-free families. The oracle for this algorithm is derived from the Hao-Orlin mincut algorithm. As such our oracle supplies posets P'_e (see the end of Section 4). The details are as follows.

Without loss of generality assume the value λ is known at the start of the centroid algorithm. The centroid algorithm runs the Hao-Orlin algorithm. Each time a maximum value preflow of value λ is found, say from $s, 1, \ldots, i-1$ to i, the Hao-Orlin algorithm is interrupted. We construct the Picard-Queyranne representation of all minimum source-sink cuts. That poset is used as the poset P'_i for the next iteration of the centroid algorithm.

The rest of the construction of $\mathcal{T}$ is straightforward: Assume we are constructing the root r of $\mathcal{T}$. If v is the element centroid at r, we construct the poset P_v labelling r by finding a maximum flow from s to v and constructing the Picard-Queyranne representation. We find the sets $M(u,v)$ corresponding to the children of r by running the Hao-Orlin algorithm again.

Theorem 8.2. For $\mathcal{F}^s$ defined for edge cuts of a network with $\lambda > 0$, $\mathcal{T}(\mathcal{F}^s)$ can be constructed in time $O(nm\log(n^2/m))$.

Proof. The correctness of the above algorithm is clear, so we discuss the time. The time to construct the root of $\mathcal{T}$ is $O(nm\log(n^2/m))$. This follows from the time bounds for the Hao-Orlin algorithm and an efficient maximum flow algorithm [GoT].

Now consider all nodes x of $\mathcal{T}$ having $n/2^{i+1} < |S_x| \leq n/2^i$, for some fixed i. Recall that an edge of G occurs in the graph induced by at most one of these sets S_x. Thus the total time spent on these nodes is at most proportional to $n/2^i \max\{\sum_{j=1}^{J} m_j \log((n/2^i)^2/m_j) \mid \sum m_j \leq m, m_j \leq (n/2^i)^2\}$. Here J is the number of the nodes. It is a simple exercise to show that the

maximum term is at most $m \log (n^2/m)$ (since the function $x \log (C/x)$ is concave downward). This implies the total time is proportional to $\sum_i (n/2^i) m \log (n^2/m) = O(nm \log (n^2/m))$. ∎

Next we give an efficient algorithm to construct the representation for a digraph (i.e., all edges have unit capacity). We begin with an observation on the Picard-Queyranne representation (that is valid for any network).

Lemma 8.2. Let dag D be the Picard-Queyranne representation of all s,t-cuts of a network G. Given G and the partition of V into the nodes of D, the dag D can be constructed in time $O(m)$.

Proof. The edges of D are constructed by taking each edge of G that joins distinct nodes of D and placing either it or its reversal (but not both) in D. (This follows since an edge of G joining two nodes of D is either saturated or void in any maximum flow. So it has residual capacity in precisely one direction.) We can determine the unknown orientation of the edges of G from a topological numbering of D. (In a topological numbering, each edge of D is directed from a lower-numbered node to a higher.) Thus it suffices to number D in topological order.

Node $[t]$ is a source of D and can be numbered 1. In general suppose we have numbered a set of nodes T. The next highest topological number can be given to any source of $D - T$. It is easy to check that a node $x \notin T$ is a source of $D - T$ iff $\rho(T \cup x) = \lambda$.

The algorithm maintains the value $\rho(T \cup x)$ for every node $x \notin T$. It initializes this value to $\rho(x) + \lambda$; when $[t]$ is added to T, the value for x is decreased by the capacity of all edges joining x and $[t]$ (in either direction). In general when a source y gets numbered and added to T, the value for x is decreased by the capacity of all edges joining x and y in either direction. Any node y with $\rho(T \cup y) = \lambda$ can be chosen as the next source. ∎

We also use the following facts relating connectivity and matroids. Given a digraph G, fix a vertex s and an integer k. A set of $k(n-1)$ edges T is a *complete k-intersection (for s)* if T contains precisely k edges directed to each vertex except s, and T can be partitioned into k spanning trees. Suppose $\mathcal{F}^s$ is nonempty. Then λ is the largest k for which G has a complete k-intersection for s [E69, E72]. A generalization of this relation states that $\mathcal{F}^s$ consists of the sets $X \subseteq V - s$ where $\rho(X) \subseteq T$ and each tree of T contains a spanning tree of X [Ga91b]. This generalization implies that a vertex v of G is an element centroid if it is an ordinary centroid of some tree of the complete intersection T. (This generalization also gives a proof independent from Theorem 2.1 that a network with rational capacities has an element centroid.)

The above relation is the basis of an algorithm that finds the connectivity λ of a digraph, plus a complete λ-intersection and its partition into λ spanning trees, in time $O(\lambda m \log(n^2/m))$ [Ga91a]. Given such a partition, [Ga91b] shows that the following two operations can be done in time $O(m)$: (i) First note that for any vertex x, the smallest set of $\mathcal{F}^s$ containing x is well-defined (it is the smallest minimum cut containing x but not s). These smallest sets can be represented by a poset. (The smallest set of $\mathcal{F}^s$ containing x is the smallest closed set containing x in the poset.) The first operation finds the partition of V into nodes of the poset. (ii) The second operation finds the maximal sets of $\mathcal{F}^s$.

We now describe the algorithm to construct $\mathcal{T}(\mathcal{F}^s)$. It starts by finding a complete λ-intersection T for s. Next it constructs each node of $\mathcal{T}(\mathcal{F}^s)$. We describe how the root is contructed, and then indicate the modifications for constructing the remaining nodes.

Choose a centroid as an ordinary centroid v of any tree of T. To construct P_v add an edge from v to every vertex x. Any set $X \subseteq V - s$ with $\rho(X) = \lambda$ in the new graph contains v. Apply the above operation (i). It finds the nodes of P_v. Then construct the poset P_v using the algorithm of Lemma 8.2.

Finally to find the sets $M(u, v)$ we use a different graph: Starting with G, add an edge sv. Any set $X \subseteq V - s$ with $\rho(X) = \lambda$ in the new graph does not contain v. Find the maximal sets of indegree λ using the above operation (ii).

The above procedure constructs the root of $\mathcal{T}(\mathcal{F}^s)$. To construct the other nodes recursively, consider a maximal $\mathcal{F}^s$-set X not containing v. As noted above, each tree of T contains a spanning tree of X. In fact adding the edges of $\rho(X)$, one per spanning tree, gives a complete intersection of the subgraph modelling $\mathcal{F}^s(X)$ (this subgraph is described in the proof of Theorem 8.1). Thus the above procedure can be applied recursively.

Theorem 8.3. For $\mathcal{F}^s$ defined for edge cuts of a digraph with $\lambda > 0$, $\mathcal{T}(\mathcal{F}^s)$ can be constructed in time $O(\lambda m \log(n^2/m))$.

Proof. We need only prove the time bound. The time to find a complete λ-intersection for s is $O(\lambda m \log(n^2/m))$. The time to construct the root of $\mathcal{T}$ is $O(m)$. Now the argument of Lemma 5.1(i) shows the time to construct all nodes of $\mathcal{T}$ is $O(m \log(n^2/m))$. ∎

Note that the centroid tree for the degree function $\mathcal{C}(\mathcal{F}, d)$ can be constructed in the same time.

Some algorithms (see Section 13 and [Ga93b]) represent $\mathcal{F}^s$ by the *node form* of $\mathcal{T}(\mathcal{F}^s)$. For any intersecting family $\mathcal{F}$, in the node form each poset label of $\mathcal{T}(\mathcal{F})$ is replaced by a list of its nodes (and their constituent vertices). The node form for $\mathcal{F}$ has size $O(n \log n)$. For the Picard-Queyranne representation, any poset label of $\mathcal{T}(\mathcal{F}^s)$ can be constructed from G and the appropriate node list, in linear time (Lemma 8.2).

9. Minimum vertex cuts

Consider a digraph G with a vertex capacity function $c : V \to \mathbf{R}_+$. Let Γ^- and Γ^+ be the neighbor functions: for $X \subseteq V$, $\Gamma^-(X)$ is the set of all vertices of $V - X$ on an edge directed to X; $\Gamma^+(X)$ is defined similarly using edges directed from X. Define $c\Gamma^-$ by $c\Gamma^-(X) = c(\Gamma^-(X))$ and similarly for $c\Gamma^+$. Thus $c\Gamma^-$ and $c\Gamma^+$ count the capacity of the neighbors of a set. It is well-known that $c\Gamma^-$ and $c\Gamma^+$ are submodular. We use these functions to derive an efficient representation for minimum vertex cuts. The special case $c \equiv 1$ is of interest, in which case the $c\Gamma$ functions count the number of neighbors.

A *vertex cut* is a partition of V into sets C, S_1 and S_2, the latter two nonempty, with no edge directed from S_1 to S_2. A *minimum vertex cut* has $c(C)$ minimum; this minimum value $c(C)$ is the *vertex connectivity* κ.

The family of minimum vertex cuts need not be intersecting or crossing. Our representation is based on a number of intersecting families of the same type. We now define this family. The family is also used in to solve the k-vertex-reachability problem in [Ga93b].

Consider a network of vertex connectivity κ. For an arbitrary vertex s define the family

$$\mathcal{F}^s = \{X \mid X \subseteq V - \Gamma^+(s) - s,\ c\Gamma^-(X) = \kappa\}.$$

$\mathcal{F}^s$ is an intersecting family since it is the family of nonempty minimizers of a submodular function. ($\mathcal{F}^s$ may be empty.)

A family $\mathcal{F}^s$ can contain triangles. In fact the family may have no element centroid, only triangle centroids. We illustrate this on undirected graphs for any connectivity κ. Form a graph by starting with K_κ and adding extra vertices s, x_i, $i = 1, \ldots, n$, each adjacent to all vertices of K_κ. This graph has vertex connectivity κ – the vertices of K_κ form unique minimum cardinality separating set. $\mathcal{F}^s$ consists of all nonempty subsets of $X = \{x_i \mid i = 1, \ldots, n\}$. Thus $\{x_1, x_2, x_3\}$ is an $\mathcal{F}^s$-triangle. Any x_i is not in the $(\mathcal{F}^s)^+$-set $X - x_i$, so x_i is not an element centroid.

We represent all minimum vertex cuts using a number of families $\mathcal{F}^s$ as follows. For a given vertex s, a minimum vertex cut S_1, S_2, C either has $s \in S_1$, which is represented by $\mathcal{F}^s$, or $s \in S_2$, which is represented symmetrically, or $s \in C$. To handle the last case let s range over all vertices

when c is arbitrary, and over any $\kappa+1$ vertices when $c \equiv 1$. Note this representation is redundant, since $\mathcal{F}^s$ can have triangles.

Now we discuss the posets for $\mathcal{T}(\mathcal{F}^s)$. We use a network that models vertex capacities. More precisely, suppose we wish to represent the minimum vertex cuts containing vertex t but not s. Construct a network having vertices $\{v^-, v^+ \mid v \in V\}$ and edges $\{v^-v^+, v^+v^- \mid v \in V\} \cup \{v^+w^- \mid vw \in E\}$; edges v^-v^+ have capacity $c(v)$ while all other edges have infinite capacity.

Consider the Picard-Queyranne representation of all minimum s^+, t^--cuts in this network. In this representation the closed set corresponding to a set $X \subseteq V - s$ with $t \in X$ and $c\Gamma^-(X) = \kappa$ contains vertices $\{x^-, x^+ \mid x \in X\} \cup \{x^+ \mid x \in \Gamma^-(X)\}$. Thus both X and its neighbors are represented. This feature increases the efficiency of the algorithm for k-vertex-reachability in [Ga93b].

In some cases the standard poset P_t gives a representation using slightly less space than if we use the Picard-Queyranne representation. To obtain P_t (the poset of $\mathcal{F}^s$-sets containing t) from the Picard-Queyranne representation, identify vertex v with the network vertex v^-; if edge v^-v^+ is in the residual graph then contract v^+ and v^- else delete v^+ (in the latter case v^+ is a sink of the Picard-Queyranne representation). Using the posets P_t it is easy to see that $\mathcal{F}^s$ has the sparse poset property. (To model the family $\mathcal{F}^s(X)$ for a set $X \subseteq V$, construct a network using the graph induced by $X \cup \Gamma^-(X)$.)

Theorem 9.1. Consider $\mathcal{F}^s$ defined for vertex cuts.

(i) $|\mathcal{T}(\mathcal{F}^s)| = O(m \log(n^2/m))$. The representation of all minimum vertex cuts has size $O(nm \log(n^2/m))$ for arbitrary c and size $O(\kappa m \log(n^2/m))$ for $c \equiv 1$.

(ii) For $c \equiv 1$, $\mathcal{T}(\mathcal{F}^s)$ can be constructed in time $O(\min\{\kappa, \sqrt{n}\}nm)$.

Proof. Part *(i)* follows from Lemma 5.1 since $\mathcal{F}$ has the sparse poset property. For part *(ii)* the time to construct a poset P_t is dominated by the time to find a maximum flow on a network with unit vertex capacities, which is $O(\min\{\kappa, \sqrt{n}\}m)$. This gives total time $O(\min\{\kappa, \sqrt{n}\}nm)$ to construct the posets at the root of $\mathcal{C}$. The hypothesis of Lemma 5.1*(ii)* does not quite hold, since constructing the posets at a node for X, $X \subseteq V$, uses time $O(\min\{\kappa, \sqrt{n_X}\}n_X m'_X)$; here m'_X is the number of edges with at least one end in X. However it is easy to see that the proof of Lemma 5.1*(ii)* still holds if we change m_X to m'_X. ∎

If we label nodes of $\mathcal{T}(\mathcal{F}^s)$ with the Picard-Queyranne representation, part *(ii)* of the theorem continues to hold. In part *(i)*, the size of $\mathcal{T}(\mathcal{F}^s)$ becomes $O(m \log n)$. (This follows since $\mathcal{C}$ has

height $\log n$.)

10. Count matroids

Matroids defined by counts on graphs arise in the study of graph rigidity and scene analysis [W]. Consider an undirected graph $G = (V, E)$. For $X \subseteq E$ let $n(X)$ denote the number of distinct vertices in the edges of X. For fixed integers a, b with $b < 2a$ define $\gamma(X) = a|n(X)| - b$. γ is a submodular function. In the *count matroid*, X is independent if $|A| \leq \gamma(A)$ for any set A with $\emptyset \subset A \subseteq X$. The special case $a = 2$, $b = 3$ is known as the rigidity matroid (recall Section 1). An independent set in the rigidity matroid contains no redundant bars, when graphs model bar-and-joint frameworks in the plane [LoY] (a bar is "redundant" if removing it does not introduce a new degree of freedom of motion).

Now suppose that for graph G, E is independent in the count matroid and $|E| = \gamma(E)$. Define the set family

$$\mathcal{F} = \{X \mid X \subseteq E,\ |X| = \gamma(X)\}.$$

$\mathcal{F}$ is an intersecting family, since it is the family of nonempty minimizers X of the function $\gamma(X) - |X|$. In the special case of the rigidity matroid the assumption on E means that graph G is a minimally rigid bar-and-joint framework ("minimally rigid" means that removing any edge from G makes it nonrigid). $\mathcal{F}$ is the set of all nonempty rigid subgraphs of G.

Lemma 10.1. $\mathcal{F}$ is a triangle-free intersecting family for $b \neq 0, a$. It is a ring family for $b = 0$.

Proof. First suppose $b \neq 0, a$. It suffices to prove the condition of Lemma 2.1(ii), i.e., any two nonempty disjoint $\mathcal{F}$-sets A and B have $A \cup B \notin \mathcal{F}$. Suppose on the contrary that A and B are disjoint $\mathcal{F}$-sets with $A \cup B \in \mathcal{F}$ ("disjoint" means the sets have no common edge, although there may be common vertices.) Then $an(A \cup B) - b = \gamma(A \cup B) = |A \cup B| = |A| + |B| = a(n(A) + n(B)) - 2b$. Equivalently, $b = a(n(A) + n(B) - n(A \cup B))$. But this is impossible since $b \neq 0, a$ and $b < 2a$.

Next suppose $b = 0$. To prove $\mathcal{F}$ is a ring family it suffices to show that the union of two disjoint $\mathcal{F}$-sets A and B is an $\mathcal{F}$-set. The number of edges in $A \cup B$ is $a(n(A) + n(B))$. By independence this number is at most $a(n(A \cup B))$. Thus $n(A) + n(B) \leq n(A \cup B)$, which implies $n(A) + n(B) = n(A \cup B)$. This implies $A \cup B \in \mathcal{F}$. ∎

As a footnote to the proof observe that in the rigidity matroid, the union of three edge-disjoint rigid graphs can be rigid, e.g., 3 triangles where 1 edge of each forms another triangle. Also observe that the case omitted in the lemma, $b = a$, is neither a ring family nor triangle-free. This is

illustrated by the graph of 3 paths with a common endpoint, with $a = b = 1$. (The $\mathcal{F}$-sets are the subtrees of the graph.)

The Centroid Theorem shows that for $b \neq 0, a$, $\mathcal{F}$ has an element centroid. This has the following interpretation in the rigidity matroid: Any minimally rigid graph has an edge whose removal leaves no rigid piece of $> m/2$ edges. Simple examples show this bound is best possible.

We turn to the representation $\mathcal{T}(\mathcal{F})$. Assume $b \neq 0$, since otherwise the ring family $\mathcal{F}$ has a trivial representation.

We first discuss the posets P_e. We use a network flow model suggested by Imai [I]. Represent the given graph $G = (V, E)$ by a graph with vertices $\{s, t\} \cup V \cup E$ and edges $\{se \mid e \in E\} \cup \{ev \mid v$ is an endpoint of $e\} \cup \{vt \mid v \in V\}$. The capacity of an edge is 1 if it is of the first type se, ∞ if it is of the second type ev, and a if it is of the third type vt. Call this network N. For any edge e the network N_e is obtained from N by increasing the capacity of edge se by b.

It is easy to check that a minimum s,t-cut has capacity $|E|$ in N, and capacity $|E| + b$ in any N_e. Furthermore a set $\{s\} \cup E' \cup V'$, where $E' \subseteq E$ and $V' \subseteq V$, has out-degree $|E| + b$ in N_e iff $e \in E'$ and $an(E') - b = |E'|$. (This uses the assumption $b > 0$.) Thus the $\mathcal{F}$-sets containing e are represented by the Picard-Queyranne representation of N_e. The size of this poset is $O(m) = O(an)$.

Theorem 10.1. Consider $\mathcal{F}$ defined for count matroids with $b \neq 0$.

(i) $|\mathcal{T}(\mathcal{F})| = O(an \log n)$.

(ii) $\mathcal{T}(\mathcal{F})$ can be constructed in time $O(b(an)^2)$.

Proof. The argument is similar to Lemma 5.1.

(i) For any integer i, consider all nodes x of $\mathcal{C}$ such that $m/2^{i+1} < |S_x| \leq m/2^i$. The total size of all posets for these nodes is $O(m)$. Thus $|\mathcal{T}(\mathcal{F})| = O(m \log n) = O(an \log n)$.

(ii) To construct $\mathcal{T}(\mathcal{F})$ first construct network N and find a maximum flow. Its value is $|E| = an - b$. Now a poset P_e is constructed by augmenting the flow to a maximum flow on N_e and then constructing the Picard-Queyranne representation.

A maximum flow on N_e is found in time $O(bm)$, since we need only find b augmenting paths. Thus the poset oracle constructs all posets P_e in time $O(bm^2)$. For a given i, the time to construct the posets of all nodes of part *(i)* is $O(bm^2/2^i)$. This gives total time $O(bm^2)$. ∎

11. Submodular flow algorithms

Our algorithms for submodular flow problems are implementations of Frank's 0-1 submodular flow algorithm [F82] (we also implement [F81], a special case of the flow algorithm). This section sketches Frank's algorithm and describes the problem that is often the bottleneck in the implementation. (A complete description of Frank's algorithm and our use of the representation in it is in [Ga93a].)

The submodular flow problem is defined on a digraph $G = (V, E)$. The 0-1 submodular flow problem asks for a maximum-weight subgraph of G satisfying certain flow constraints. The algorithm does a sequence of $\leq m$ "augmentations." Each of these amounts to a shortest-path calculation on a working graph H. H contains certain artificial "jumping edges." Constructing the jumping edges is often the dominating part of the computation (e.g., this holds for the algorithms discussed here and in [Ga93b]).

We model the construction of jumping edges using the problem of Section 6: Let $\mathcal{F}$ be the family of all sets $S \subseteq V$ where the current flow constraint is "tight." $\mathcal{F}$ is an intersecting or crossing family. Constructing H amounts to answering a sequence of intersection queries $\mu(v, \mathcal{F})$ for each vertex v. (We do not construct all jumping edges at once, as in [F82]. Rather when a vertex v is reached in the search, we construct the jumping edges from v using a query $\mu(v)$. This organization limits the space for H to $O(m)$, rather than $O(n^2)$ if all jumping edges were constructed at the start.)

12. Dijoins

Consider a digraph $G = (V, E)$. For a set of vertices S, $\emptyset \subset S \subset V$, the set of edges directed from $V - S$ to S is a *dicut* if $\delta(S) = 0$. A *dijoin* is a set of edges that meets every dicut. Note that a weakly connected digraph can be made strongly connected by contracting (or adding the reverse of) every edge of a dijoin. The *minimum-cost dijoin problem* is, given a digraph with edge-cost function $c : E \to \mathbf{R}_+$, find a dijoin with smallest possible total cost. (To put this in perspective recall that it is NP-complete to make a digraph strongly connected by adding the fewest number of edges chosen from a given set E' [ET].) The Lucchesi-Younger Theorem gives a minimax characterization of the solution [LY, GLS]. Frank gives an algorithm that finds a minimum-cost dijoin in time $O(n^3 m)$ using $O(m)$ space, or $O(n^2 M(n))$ time using $O(n^3)$ space [F81].

For the dijoin problem Frank's algorithm does $\leq n$ augmentations. Now we describe the family that defines the jumping edges. The algorithm maintains a dijoin D. The family defining jumping edges is

$$\mathcal{F} = \{X \mid X \subseteq V,\ \delta(X) = 0,\ \rho_D(X) = 1\}.$$

Here δ is the out-degree function for G and ρ_D is the in-degree function for the dijoin D currently constructed by the algorithm. $\mathcal{F}$ is a crossing family. (Note $\mu(v)$ need not be an $\mathcal{F}$-set since possibly $\rho_D(\mu(v)) > 1$.) Our definition of $\mu(v)$ is equivalent to [F81] by proposition (5.1) of [F81]. [F81] finds all sets $\mu(v)$, $v \in V$, in time $O(n^2m)$ or $O(nM(n))$. We reduce this by a factor of n, achieving the following.

Theorem 12.1. A minimum-cost dijoin can be found in $O(n^2m)$ time using $O(m)$ space, or alternatively $O(nM(n))$ time using $O(n^2)$ space.

Proof. Define a multidigraph G_D by starting with G, adding another copy of each edge of E, and adding the reverse of each edge of D. The $\mathcal{F}$-sets are precisely the vertex sets of out-degree one in G_D.

Construct the centroid trees for the degree function $\mathcal{C}(\mathcal{F}^s, d)$ and $\mathcal{C}(\mathcal{F}_s, d)$. This uses time $O(m \log(n^2/m))$ (by Theorem 8.3 with $\lambda = 1$). A poset P_e can be constructed in $O(m)$ time, using the Picard-Queyranne representation on graph G_D. Thus Lemma 6.4(i) shows that a set $\mu(v)$ can be found in time $O(m)$. This gives time $O(nm)$ to find all sets $\mu(v)$. Since there are $\leq n$ augmentations, the total time is $O(n^2m)$. The space is $O(m)$.

An alternate implementation constructs the representations $\mathcal{T}(\mathcal{F}^s)$ and $\mathcal{T}(\mathcal{F}_s)$, and converts them to transitively-closed forms $\mathcal{T}^c(\mathcal{F}^s)$ and $\mathcal{T}^c(\mathcal{F}_s)$. The time to convert is $O(M(n))$ and the space for these representations is $O(n^2)$ (see Section 3). The time to find all sets $\mu(v)$ is $O(n^2)$, by Lemma 6.2(i). This gives total time $O(nM(n))$ and space $O(n^2)$. ∎

An application of the dijoin problem is finding a minimum-cost feedback arc set for a planar digraph. Feedback arc sets correspond to dijoins in the dual graph (since directed cycles correspond to dicuts) [GLS].

Corollary 12.1. A minimum-cost feedback arc set in a planar digraph can be found in $O(n^3)$ time and $O(n)$ space. ∎

13. Connectivity orientation

An *orientation* of an undirected graph assigns a direction to each edge. The *k-edge-connected orientation problem* is to orient a given undirected graph G to make it k-edge-connected or show this is impossible. Nash-Williams gives a minimax characterization of the solution [NasW]. In the

minimum-cost k-edge-connected orientation problem each orientation of an edge has a cost and we seek a k-edge-connected orientation of smallest possible cost.

[F82] shows that a minimum-cost orientation can be found in two steps: First find a k-edge-connected orientation. Then apply Frank's submodular flow algorithm to convert the orientation to one with minimum cost. We begin by discussing this conversion step.

The conversion step does $\leq kn$ augmentations. The family that defines the jumping edges is the family of minimum edge cuts in a graph G'. G' is an orientation of the given graph G having edge-connectivity k.

Theorem 13.1. Given a k-edge-connected orientation, a minimum-cost k-edge-connected orientation can be found in $O(kn^2m)$ time using $O(m)$ space, or alternatively $O(kn(km\log(n^2/m) + M(n)))$ time using $O(n^2)$ space.

Proof. Our implementation of the conversion step is similar to the dijoin problem. We use the representation of minimum edge cuts for G'. Theorem 8.3 shows the representation is found in time $O(km\log(n^2/m))$.

Suppose (as in Theorem 12.1) we use centroid trees for the degree function $\mathcal{C}(\mathcal{F}^s, d)$ and $\mathcal{C}(\mathcal{F}_s, d)$. In addition we store the two complete k-intersections for G' used to construct these representations. A poset P_e is constructed in $O(m)$ time (as in the algorithm for constructing the representation). Thus Lemma 6.4(i) shows that each augmentation uses time $O(nm)$. (Since $k \leq n$, this dominates the initial time to construct the representation.) Since there are $\leq kn$ augmentations, the total time is $O(kn^2m)$. The space is $O(m)$.

Using transitively-closed representations as in Theorem 12.1 gives the second bound of the theorem. ∎

It remains to solve the k-edge-connected orientation problem. This is done in [Ga93a] in time dominated by Theorem 13.1. Thus a minimum-cost k-edge-connected orientation can be found in the time bounds of Theorem 13.1. [Ga93b] improves the time to find a k-edge-connected orientation by extending [Ga93a] with the node form of the representation.

Acknowledgments. We thank András Frank and Éva Tardos for their advice.

References

[CW] D. Coppersmith and S. Winograd, "Matrix multiplication via arithmetic progressions," *J. Symbolic Comp. 9*, 1990, pp. 251–280.

[DKL] E.A. Dinits, A.V. Karzanov and M.V. Lomonosov, "On the structure of a family of minimal weighted cuts in a graph", in *Studies in Discrete Optimization*, A.A. Fridman (Ed.), Nauka Publ., Moscow, 1976, pp. 290–306.

[E69] J. Edmonds, "Submodular functions, matroids, and certain polyhedra", *Calgary International Conf. on Combinatorial Structures and their Applications*, Gordon and Breach, New York, 1969, pp. 69–87.

[E72] J. Edmonds, "Edge-disjoint branchings", in *Combinatorial Algorithms*, R. Rustin, Ed., Algorithmics Press, New York, 1972, pp. 91–96.

[Ev] S. Even, *Graph Algorithms,* Computer Science Press, Potomac, MD, 1979.

[ET] K.P. Eswaran and R.E. Tarjan, "Augmentation problems", *SIAM J. Comput., 5,* 4, 1976, pp. 653–665.

[F78] A. Frank, "On disjoint trees and arborescences", in *Algebraic Methods in Graph Theory*, L. Lovász and V.T. Sós, Eds., Colloq. Math. Soc. János Bolyai, 18, North-Holland, New York, 1978, pp. 159–169.

[F81] A. Frank, "How to make a digraph strongly connected", *Combinatorica 1*, 2, 1981, pp. 145–153.

[F82] A. Frank, "An algorithm for submodular functions on graphs", in *Bonn Workshop on Combinatorial Optimization*, A Bachem, M. Grötschel and B. Korte, Eds., Ann. Discrete Math., 16, North-Holland, Amsterdam, 1982, pp. 97–120.

[F84] A. Frank, "Finding feasible vectors of Edmonds-Giles polyhedra", *J. Comb. Th., B, 36*, 1984, pp. 221–239.

[FT] A. Frank and É. Tardos, "An application of submodular flows", *Linear Algebra and Its Applications, 114/115*, 1989, pp. 329–348.

[Fu90] S. Fujishige, *Submodular Functions and Optimization*, North-Holland, New York, 1990.

[Ga91a] H.N. Gabow, "A matroid approach to finding edge connectivity and packing arborescences", *Proc. 23rd Annual ACM Symp. on Theory of Comp.,* 1991, pp. 112–122.

[Ga91b] H.N. Gabow, "Applications of a poset representation to edge connectivity and graph rigidity", *Proc. 32nd Annual Symp. on Found. of Comp. Sci.*, 1991, pp. 812–821.

[Ga93a] H.N. Gabow, "A framework for cost-scaling algorithms for submodular flow problems", *Proc. 34th Annual Symp. on Found. of Comp. Sci.*, 1993, to appear.

[Ga93b] H.N. Gabow, "Efficient algorithms for submodular flow problems on intersecting families", in preparation.

[GLS] M. Grötschel, L. Lovász and A. Schrijver, *Geometric Algorithms and Combinatorial Optimization*, Springer-Verlag, New York, 1988.

[GoT] A.V. Goldberg and R.E. Tarjan, "A new approach to the maximum flow-problem", *J. ACM, 35,* 4, 1988, pp. 921–940.

[H] F. Harary, *Graph Theory*, Addison-Wesley, Reading, Mass., 1969.

[HO] J. Hao and J.B. Orlin, "A faster algorithm for finding the minimum cut in a graph", *Proc. 3rd Annual ACM-SIAM Symp. on Disc. Algorithms,* 1992, pp. 165–174.

[I] H. Imai, "Network-flow algorithms for lower-truncated transversal polymatroids", *J. Op. Res. Soc. of Japan, 26*, 3, 1983, pp. 186–210.

[Kar] A.V. Karzanov, "On the minimal number of arcs of a digraph meeting all its directed cutsets", abstract, *Graph Theory Newsletters, 8*, March 1979.

[Kan] A. Kanevsky, "On the number of minimum size separating vertex sets in a graph and how to find all of them", *Proc. 1st Annual ACM-SIAM Symp. on Disc. Algorithms,* 1990, pp. 411–421.

[Lu] C.L. Lucchesi, "A minimax equality for directed graphs", Ph. D. Dissertation, University of Waterloo, Waterloo, Ontario, 1976.

[LY] C.L. Lucchesi and D.H. Younger, "A minimax relation for directed graphs", *J. London Math. Soc., 2,* 17, 1978, pp. 369–374.

[LoY] L. Lovász and Y. Yemini, "On generic rigidity in the plane", *SIAM J. Alg. Disc. Meth., 3,* 1982, pp. 91–98.

[N] M. Nakamura, "On the representation of the rigid sub-systems of a plane link system", *J. Op. Res. Soc. of Japan, 29*, 4, 1986, pp. 305–318.

[NasW] C. St. J. A. Nash-Williams, "Well-balanced orientations of finite graphs and unobtrusive odd-vertex-pairings", in *Recent Progress in Combinatorics*, W.T. Tutte, Ed., Academic Press, New York, 1969, pp. 133–149.

[PQ] J.-C. Picard and M. Queyranne, "On the structure of all minimum cuts in a network and applications", *Math. Prog. Study 13*, 1980, pp. 8–16.

[R] A. Recski, *Matroid Theory and its Applications in Electric Network Theory and in Statics*, Springer-Verlag, New York, 1989.

[T] W.T. Tutte, "On the problem of decomposing a graph into n connected factors", *J. London Math. Soc., 36*, 1961, pp. 221–230.

[W] W. Whiteley, "A matroid on hypergraphs, with applications in scene analysis and geometry", *Discr. Comp. Geom. 4*, 1989, pp. 75–95.

[WW] R.J. Wilson and J.J. Watkins, *Graphs: An Introductory Approach*, John Wiley and Sons, New York, 1990.